Date of publication MAR-31, 2025, date of current version JAN-08, 2025.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.24.1.3883

Peculiarities of Adaptation of the LZ77 Dictionary Algorithm to Lossless Image Compression

ALEXANDER SHPORTKO[1], ANDRII BOMBA[2]
[1]Department of Information Systems and Computing Methods, Academician Stepan Demianchuk International University of Economics and Humanities, 4, Acad. S. Demianchuk Str, 33000, Rivne, Ukraine
[2]Department of Computer Sciences and Applied Mathematics, National University of Water and Environmental Engineering, 11, Soborna Str, 33028, Rivne, Ukraine

Corresponding author: Alexander Shportko (e-mail: ITShportko@gmail.com).

ABSTRACT The article describes three modifications of the LZ77 dictionary algorithm for lossless image compression in the process of sequential bypass pixels: storing offsets to identical sequences in pixels, not in individual components; performing a search for identical sequences first, starting from adjacent previously processed pixels; performing a search for identical sequences not only in the horizontal but also in the vertical direction. The first of these modifications is shown to improve compression by using smaller values for storing offsets and increasing the dictionary of the LZ77 algorithm threefold. The second modification forms small offsets to adjacent pixels of the previous row. And the third one finds longer identical sequences. Storing offsets in pixels and separately searching for identical sequences from adjacent processed pixels is recommended to use in graphic formats, as they improve compression with almost no impact on encoding and decoding time. The search for identical sequences in two directions is suggested to be used only in archivers, because the implementation of this modification slows down both encoding and decoding, improving the compression of only individual images. On the well-known ACT test set, it is shown that the application of the proposed modifications together with the simultaneous search of the same sequences in three dictionaries makes it possible to reduce image compression coefficients by an average of 0.18 bpb.

KEYWORDS lossless image compression; dictionary compression methods; LZ77 algorithm schedules.

I. INTRODUCTION

TODAY, images are an inseparable component of multimedia information enhancing our understanding of the world around us. Compressing images speeds up their transfer over the network and reduces disk space usage. All graphic formats that perform image data compression implement methods of one of two compression classes: lossy (for example, JPEG [1, 2]) and lossless (for example, PNG [2, 3]). And if for most lossy image compression algorithms it is possible to achieve the required compression ratio (hereafter CR, volume ratio compressed to uncompressed image file sizes, expressed in bpb) at the expense of quality degradation, then the compression level of lossless images is on average only 30-70%, it depends on pixel color gradients and the compression algorithm itself and cannot be programmed. Therefore, the problem of improving the efficiency of lossless image compression is relevant at this time and will remain the same in the future.

II. FORMULATION OF THE PROBLEM

Usually, the process of lossless image compression in archivers and graphic formats can be divided into a maximum of four stages: at the first stage, context-dependent coding reduces inter-element redundancy between the same fragments or fragments with the same structure; at the second stage, the transition to an alternative color model is performed [4]; at the third – instead of the brightness of the raw pixel components, their deviations from the values predicted by the predictors are stored (forecaster) [5-7]:

$$\Delta_{uv} = brightness_{uv} - predict_{uv} \qquad (1)$$

(u and v are changing, respectively, along all the rows and columns of the pixel components of the image). At the fourth stage, context-independent coding handles code redundancy by forming element codes with lengths dependent on their probabilities (for example, using Huffman codes [8-11] or arithmetic codes [12-15]). The average length of such a code approaches entropy [16] is as follows:

$$H = -\sum_i p_i \times \log p_i \quad , \qquad (2)$$

where p_i is the probability of the i-th element. The second and third stages redistribute the luminance values of the components without directly performing compression, but with this they increase the unevenness of the distribution of values and therefore increase the efficiency of the fourth stage of coding. It is possible to additionally increase the unevenness of the distribution of elements by switching to color models used in publishing, for example, to the HSI model [17], which is invariant to changes in lighting. However, while switching to such models and returning from them to the RGB model, the operations of dividing the component values are repeatedly performed, which is unacceptable for lossless image compression. Therefore, at the second stage of such compression, difference color models are used, in which a maximum of two components are replaced by differences with other components [4]. These color models provide unambiguous decoding.

One of the successful examples of the combination of context-dependent and context-independent stages of coding, approved at the standard level, is the Deflate dictionary compression format [18]. This format uses the context-dependent dictionary algorithm LZ77 [19-22] to process the input stream, and the results of its work are compressed by Huffman dynamic codes [8] to ensure the smallest CR. Today, this format is also used in many popular archivers (for example, GZIP [23]) and graphic formats (for example, PNG [3]) in other application software and does not require the purchase of licenses. In this article, we propose the modifications to the LZ77 deflate algorithm based on the PNG graphic format.

It should be recollected that the LZ77 algorithm is based on the replacement during encoding in the output stream of the same sequence for *the buffer* by reference to the same sequence starting *in the dictionary* in the form of a pair of numbers *<length of the same sequence, displacement from the end of the dictionary to the previous identical sequence>*. If it is not possible to find the same sequence in the dictionary for elements from the beginning of the buffer, then the first element (*literal)* from the beginning of the buffer is written to the end of the dictionary and to the output stream without changes, and the coding continues similarly from the next element of the buffer [19].

When decoding LZ77 codes, literals are passed to the output stream unchanged. Pairs *<length*; *offsets>* are decoded by sequentially transferring the specified number of literals from the end of the output stream, at the given offset, into the output stream.

In the RGB color model the order of sequential bypass of the brightness of the pixel components is conventionally shown by a continuous arrow in Fig. 1. The same sequence is used to form the input stream for image compression by the LZ77 algorithm in PNG format.

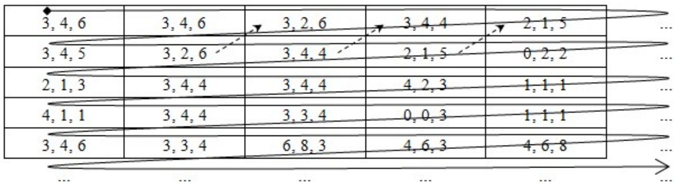

Figure 1. Brightness components of the upper pixels of the left corner of conditional RGB image and sequence of their consistent bypass (defined by continuous arrow)

For example, during a sequential bypass of the first 15 the brightness components of four pixels from Fig. 1, a stream *3, 4, 6, 3, 4, 6, 3, 2, 6, 3, 4* is formed. The LZ77 algorithm will convert this stream into a sequence *3, 4, 6, <4, 3>, 2, <3, 6>*. The step-by-step results of using of the LZ77 algorithm before changes the dictionary and buffer for this flow are shown in Table 1.

It is clear that the more and the longer the replacement of luminances in pairs *<length; offset>* is found – the lower the compression ratio will be provided by the LZ77 algorithm. Known modifications of the LZ77 algorithm and the results of their application for lossless image compression are given in [24].

Table 1. Step-by-step results of forming the schedule of the LZ77 algorithm *3, 4, 6, 3, 4, 6, 3, 2, 6, 3, 4* according to [8]

Sliding window (input stream)		The same sequence	Encoded data (output stream)	
dictionary	buffer		<length, offset>	element
-	3, 4, 6, 3, 4, 6, 3, 2, 6, 3, 4	-	-	3
3	4, 6, 3, 4, 6, 3, 2, 6, 3, 4	-	-	4
3, 4	6, 3, 4, 6, 3, 2, 6, 3, 4	-	-	6
3, 4, 6	3, 4, 6, 3, 2, 6, 3, 4	3, 4, 6, 3	<4, 3>	-
3, 4, 6, 3, 4, 6, 3	2, 6, 3, 4	-	-	2
3, 4, 6, 3, 4, 6, 3, 2	6, 3, 4	6, 3, 4	<3, 6>	-
3, 4, 6, 3, 4, 6, 3, 2, 6, 3, 4		-	-	-

To increase the number of replacements of literals and reduce the CR in the process of sequentially traversing pixels, we proposed in [25] a modification of the LZ77 algorithm, which was named LZPR. In this modification, the same sequences of elements are searched not only in the sliding window of the image data, but also in two additional sliding windows of the results of the *LeftPredict* and *AbovePredict* predictors, i.e., the search is performed not only among the same brightness component of the pixels, but also among the same brightness increments in two directions. At the same time, replacements of literals are stored in the form of a triplet of numbers *<length; displacement; sliding window number>*. If it is not possible to find identical sequences in three dictionaries, then the LZPR algorithm transfers the result of applying the predictor with the lowest entropy to the coded data instead of the next element. For example, for the input stream from Fig. 2 the lowest entropy is provided by the *AbovePredict* predictor*,* so its values are written to the output stream instead of elements that are not included in the substitution *<length; displacement>*. Thus, the LZPR algorithm finds more identical sequences of elements and forms an output stream with a lower entropy (2) relative to the classic LZ77 algorithm.

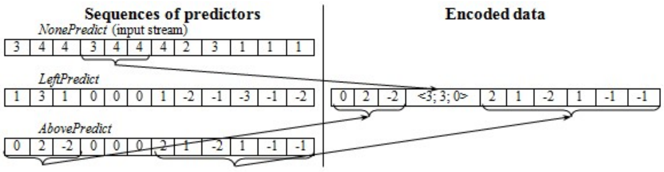

Figure 2. Application of the LZPR algorithm, starting from the second pixel in the third line of conditional RGB image shown in Figure 1

Therefore, the purpose of this article and the novelty of the research is to improve the LZ77 and LZPR algorithms due to more compact storage of displacements, increasing the dictionary and searching for identical sequences not only in the horizontal direction, but also in the vertical direction and from the nearest processed pixels to reduce the CR of images without losses in the process of sequentially traversing pixels.

III. THE WAYS TO IMPROVE COMPRESSION WITH THE LZ77 ALGORITHM DURING LOSSLESS IMAGE COMPRESSION

Firstly, we will show that the displacement of the modified LZ77 algorithm should be determined in pixels, and not in the brightness of individual components, as was done in [25]. It is well known that the highest correlation in images is between identical components of adjacent pixels [26]. This is what determines the mechanism of using predictors (1) to reduce entropy (2), which are used component by component. It is clear that different sequences of components after the application of linear predictors can generate the same brightness increments, but due to noise exposure this happens much less often than identical fragments of images. That is, it is expedient to search for identical fragments starting from the same component of previously processed pixels (if, for example, the next literal of the dictionary contains the value of component *R,* then it is expedient to search for identical sequences starting from component *R* of the processed pixels). Offsets between identical pixel components are multiples of pixel length, so these offsets can be defined in pixels rather than components. In the RGB color model that we use to store uncompressed images, brightness of pixels is represented by three components. Therefore, the displacements expressed in pixels will be three times smaller than the values of the same displacements recorded in the components, and smaller displacement values in the Deflate format are on average encoded with a smaller number of bits [17]. In addition, moving from setting offsets in components to specifying offsets in pixels triples the size of the dictionary. Therefore, such transition should predictably reduce CR.

Secondly, let us point out the expediency of encoding displacements to the nearest previously processed pixels with smaller values than to the pixels in the dictionary. Since adjacent pixels have the highest level of correlation in images, the identical luminances of pixel sequences are most often found in images starting from adjacent pixels. By their nature, images are two-dimensional, while LZ77 dictionary algorithm is text-oriented and therefore one-dimensional. Sequential pixel traversal converts a two-dimensional image into a one-dimensional stream suitable for processing by the LZ77 algorithm. But at the same time, the displacements in the dictionary to adjacent pixels in the previous row become significantly larger than the displacement to the processed pixel on the left (Fig. 3), which negatively affects the CR, since larger displacements in the Deflate format [18] are encoded with more additional bits.

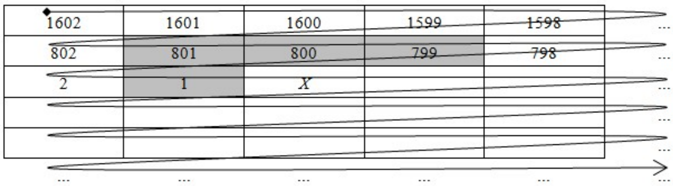

Figure 3. Offset to the adjacent pixels in the LZ77 algorithm in the dictionary (on the gray background) for pixel *X* during sequential traversal for an 800-pixel-wide image

Therefore, we encode the displacement to the four closest previously processed pixels with the smallest values from 1 to 4, and the displacement in the dictionary is increased by 4 for decoding clarity (Fig. 4). The codes of the nearest four pixels can be rearranged among themselves and this will not affect the CR, because these codes in the Deflate format are stored without additional bits. The reduction in CR here is planned to be achieved by reducing the displacements to adjacent pixels in the previous row relative to the displacements to them in the dictionary, because the pixel on the left has a small displacement. In addition, searching for the same sequences separately, starting from the nearest previously processed pixels, speeds up further searches in the dictionary, because only longer identical sequences are then searched in it.

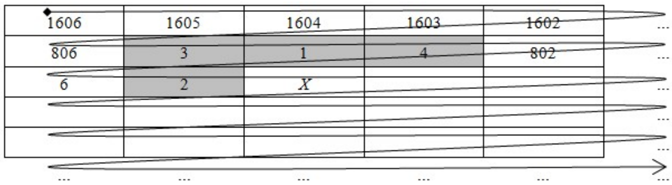

Figure 4. Codes of displacements to adjacent pixels in the LZ77 algorithm (on the gray background) for coding the same sequences on the horizontal in the process of consistent detour

Thirdly, we note that identical sequences from adjacent previously processed pixels can be searched in images not only in horizontal, but also in vertical directions, because the image is two-dimensional, and the longest found sequence from two directions can be encoded. For example, from the second pixel in the second row in Fig. 5, it is advisable to encode the same sequence with a length of 4 pixels in the vertical direction, because in the horizontal direction from this pixel only 3 pixels are the same. At the same time, coded pixels should be marked vertically in a separate array to avoid their re-coding in the process of subsequent sequential traversal.

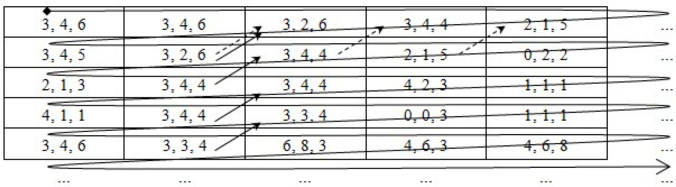

Figure 5. Search for the same sequences, beginning with the nearest processed pixels, both in the horizontal direction (indicated by dotted arrows) and in the vertical direction (indicated by solid arrows).

During reproduction of encoded images, the decoder must know the direction of continuation of each identical sequence. Therefore, in this case, the closest previously processed pixels will be coded as in Fig. 6, and increase the offset to the pixels in the dictionary by 8. For example, the same sequence of four pixels in the vertical direction from the second pixel in the second row in Fig. 5 is then encoded by a pair of numbers <4; 8>.

1610	1609	1608	1607	1606	...
810	5→, 6↓	1→, 2↓	7→, 8↓	806	...
10	3→, 4↓	*X*			...
					...
					...
...	...	...	...	...	...

Figure 6. Offset codes to the nearest previously processed pixels in the LZ77 algorithm (on the gray background) for coding the same sequences horizontally and vertically in the process of sequential traversal

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We will now demonstrate the effect of the transition from setting displacements in components to determining displacements in pixels on the CR of images using the popular ACT test set (Fig. 7). We can download these images from [27]. This set contains both synthesized (№№. 1, 2, 7) and photorealistic (all others) images. Compression ratio of the images of this set during sequential traversal in a modified PNG format with different options for determining the displacements of the LZ77 algorithm replacements are shown in Table 2. We made modifications to the MinPNG utility [28], which currently provides minimal CR in this format. Comparing the first two rows of this table, we can see that limiting the displacements to multiple pixels only increases the average CR of the ACT set images by 0.04 bpb at the expense of photorealistic images, for which sequences of the same brightness increments occur from different components.

The determination of displacements in pixels instead of displacements in components (first and third rows in Table 2) provided a reduction in the CR of the images of the ACT set by an average of 0.03 bpb, both at the expense of photorealistic and synthesized images. Such a transition did not worsen the CR of any image, so we can use it in the future. In addition, definition of offsets in pixels in the Deflate format [18] made it possible to increase the size of the dictionary to 32768 pixels, i.e., to 98304 literals. The use of such a triple dictionary (the first lines of Table 3 – Table 5) increased the probability of finding the same sequences for the elements of the buffer and therefore, on average, additionally reduced the CR by 0.02 bpb, although it slowed down the encoding by 54.5%. Therefore, storing the offsets of the LZ77 algorithm in pixels made it possible to reduce the CR on the ACT set during sequential traversal by an average of 0.05 bpb.

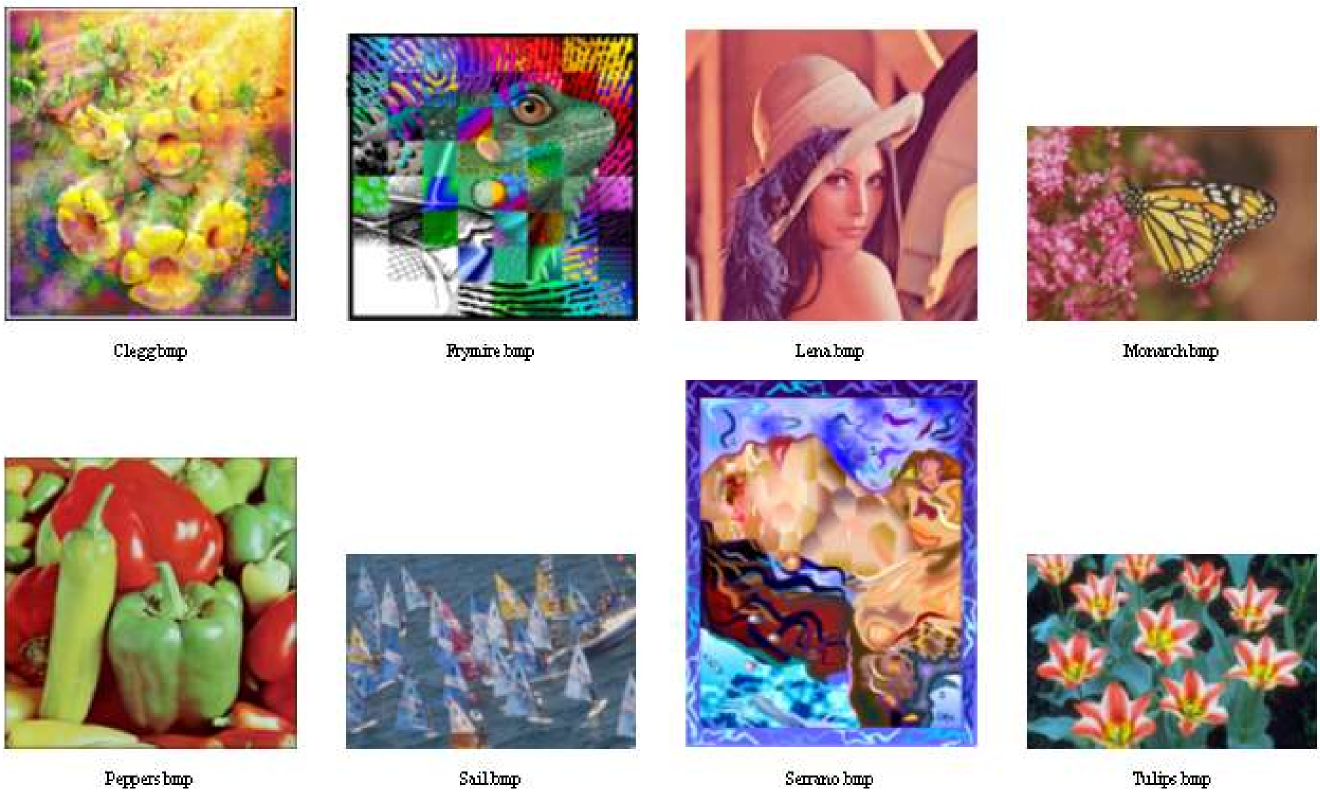

Figure 7. Images of the ACT set

Table 2. Image compression coefficients of the ACT set for different methods of determining the displacements of the LZ77 algorithm while traversing sequential pixel traversal, bpb

Modification of the LZ77 algorithm	File number								Average CR
	1	2	3	4	5	6	7	8	
Definition of all displacements of the LZ77 algorithm in components	1.67	0.50	4.82	4.13	4.30	5.16	0.52	4.59	3.21
Definition of displacements of the LZ77 algorithm is done only in components that are multiples of pixels	1.67	0.50	4.82	4.16	4.31	5.38	0.52	4.61	3.25
Definition of displacements of the LZ77 algorithm in pixels according to the standard dictionary	1.67	0.47	4.82	4.09	4.30	5.03	0.49	4.59	3.18

We will also examine the results of applying the proposed modifications to the LZPR algorithm [25] on the same test set of images (Table 3 – Table 5). We can see that, in addition to the sliding window on the image data, the application of two additional sliding windows on the data of the linear predictors *LeftPredict* and *AbovePredict* in the LZPR algorithm compared to the modified PNG format with the displacements of the LZ77 algorithm in pixels according to the triple dictionary makes it possible to reduce the CR on average for the ACT set by 0.12 bpb, accelerates encoding by 5.39 times, as well as slows down decoding by 31.1% (the first two lines of these tables), which indicates the feasibility of using the LZPR algorithm.

Table 3. Image compression coefficients of the ACT set by various modifications of the LZPR algorithm in the process of sequentially traversing pixels, bpb

Modification of the LZPR algorithm	File no								Average CR
	1	2	3	4	5	6	7	8	
MinPNG with triple dictionary pixel offsets (for comparison)	1.67	0.46	4.82	4.07	4.29	4.95	0.48	4.56	3.16
"Greedy" LZPR decomposition over three dictionaries	1.61	0.45	4.84	3.78	4.10	4.69	0.48	4.38	3.04
"Greedy" LZPR schedule by the nearest processed pixels	1.64	0.63	4.85	3.91	4.20	5.36	0.63	4.42	3.21
LZPR by nearest processed pixels and three dictionaries	1.60	0.44	4.84	3.78	4.09	4.69	0.46	4.38	3.03
LZPR by nearest processed pixels and image dictionary	1.59	0.46	4.85	3.89	4.21	5.09	0.49	4.45	3.13
LZPR with differential color models	1.60	0.44	4.70	3.37	3.71	4.11	0.46	3.77	2.77
LZPR in two directions	1.33	0.46	4.84	3.78	4.09	4.69	0.49	4.38	3.01
LZPR in two directions with different color patterns	1.33	0.46	4.70	3.37	3.70	4.11	0.49	3.77	2.74

Table 4. The time of encoding images of the ACT set using various modifications of the LZPR algorithm in the process of sequentially traversing pixels, sec

Modification of the LZPR algorithm	File no								Average time
	1	2	3	4	5	6	7	8	
MinPNG with triple dictionary pixel offsets (for comparison)	10.75	20.48	4.94	11.84	7.25	7.08	7.81	10.10	10.03
"Greedy" LZPR decomposition over three dictionaries	1.67	2.47	1.19	2.47	1.53	2.44	1.00	2.14	1.86
"Greedy" LZPR schedule by the nearest processed pixels	1.46	1.61	0.61	0.95	0.59	0.83	0.70	1.11	0.98
LZPR by nearest processed pixels and three dictionaries	1.77	2.52	1.30	2.47	1.56	2.50	1.11	2.24	1.93
LZPR by nearest processed pixels and image dictionary	1.14	1.63	0.59	0.86	0.63	1.09	0.64	0.91	0.94
LZPR with differential color models	1.72	2.81	1.31	2.55	1.63	2.72	1.23	2.22	2.02
LZPR in two directions	1.77	2.89	1.31	2.72	1.72	2.70	1.28	2.44	2.10
LZPR in two directions with different color patterns	1.81	3.28	1.34	2.78	1.74	2.86	1.41	2.45	2.21

Table 5. The decoding time of the ACT set images encoded by various modifications of the LZPR algorithm in the process of sequential pixel traversal, sec

Modification of the LZPR algorithm	File no								Average time
	1	2	3	4	5	6	7	8	
MinPNG with triple dictionary pixel offsets (for comparison)	0.64	0.70	0.30	0.45	0.36	0.48	0.22	0.47	0.45
"Greedy" LZPR decomposition over three dictionaries	0.70	0.94	0.45	0.58	0.44	0.61	0.36	0.64	0.59
"Greedy" LZPR schedule by the nearest processed pixels	0.59	0.91	0.39	0.48	0.30	0.58	0.36	0.45	0.51
LZPR by nearest processed pixels and three dictionaries	0.58	0.86	0.38	0.49	0.28	0.52	0.34	0.44	0.49
LZPR by nearest processed pixels and image dictionary	0.58	0.89	0.33	0.45	0.36	0.45	0.30	0.55	0.49
LZPR with differential color models	0.52	0.88	0.31	0.52	0.28	0.49	0.30	0.42	0.47
LZPR in two directions	0.59	1.02	0.36	0.49	0.34	0.47	0.36	0.53	0.52
LZPR in two directions with different color patterns	0.55	0.97	0.36	0.48	0.38	0.47	0.34	0.53	0.51

Searching for identical sequences, only from the closest previously processed pixels (that is, using only the four smallest offsets marked in Fig. 4), worsens the CR on average over the ACT set by 0.17 bpb, since it excludes other dictionary positions from consideration, but speeds up the encoding by almost 2 times and decoding by 13.6% (third lines of Table 3 – Table 5).

If the search for the same sequences is performed first from the nearest previously processed pixels, and then in three dictionaries, then the CR on average for the ACT set will additionally decrease by 0.01 bpb, and the duration of encoding and decoding will not undergo significant changes. At the same time, a decrease in CR is observed for all synthesized and only for one photorealistic image, since the images of the first type have more identical fragments or fragments with the same increments, and, accordingly, more substitutions of the LZ77 algorithm, for which the reduction of displacements to adjacent previously processed pixels reduces the size of the compressed file.

Using of only one image dictionary instead of three dictionaries increases the CR by an average of 0.1 bpb (fifth row of Table 3), but for images of different types this indicator worsens unequally: for synthesized images, the CR increased on average by only 0.013 bpb, and for photorealistic ones – up to 0.16 bpb. That is, a dictionary of image data is most often used to compress synthesized images, and for photorealistic images, dictionaries of predictor results (increase in horizontal and vertical directions, analogues of delta coding [29]) are also used. The same trend can be traced after applying the difference color models described in [4] – on average, the CR of ACT set decreases by 0.25-0.26 bpb mainly due to photorealistic images.

Now let us analyze the impact on the compression indicators of the search for identical sequences of the LZPR algorithm from the nearest processed pixels not only in horizontal, but also in vertical directions (fourth and seventh lines of Table 3 – Table 5). We can see that even though the ACT set has reduced the CR on average by 0.02 bpb, the improvement in compression is observed only for one synthesized image, and the deterioration is observed for two. This happens because individual identical sequences found vertically can reduce or even eliminate subsequent identical sequences horizontally, or can be completely included in such sequences. In addition, to store in the compressed data identical vertical sequences from the nearest previously processed pixels, small offsets are used (in our case, 2, 4, 6, 8, see Fig. 6), which, in addition to the Huffman code, are coded with a maximum of one additional bit [18]. Without searching for identical sequences vertically, four additional small offsets would be applied to encode identical sequences in the dictionary. All this negatively affects the CR of images. On the other hand, the CR of image #1 Clegg.bmp due to the application of an additional search for identical sequences vertically decreased by as much as 0.27 bpb, and when applying two additional iterations to discard ineffective replacements of the LZPR algorithm [25]. This image is compressed to 347 Kb, which repeats the best results of compressing this image among the archivers and graphic formats known to us. This result is also due to the fact that there is a black and white frame around the image of this file (Fig. 7). It is clear that the horizontal lines of these uniform frames are better encoded by long identical sequences horizontally, and the vertical lines by identical sequences vertically. The CR of photorealistic images was almost unaffected by the additional search for identical sequences vertically, since they contain few identical sequences.

Image encoding time due to the additional search for identical vertical sequences for the ACT set increased by 8.1% on average, and decoding time by 6.12% (see the fourth and seventh rows in Table 4 – Table 5). All this in combination with the need to use an additional array of pixel belonging to previous replacements in the coding process indicates the impracticality of implementing an additional search for the same vertical sequences for the LZ77 algorithm and its modifications in graphic formats. Such an additional search should be performed in archivers to ensure improved image compression.

V. DISCUSSION

We use the modifications of the LZ77 and LZPR algorithms described in this work (determination of offsets to the same sequences in pixels and performing an additional search for the same sequences, starting from the nearest previously processed pixels) in the graphic format of progressive hierarchical image compression without losses HBF-LS [30]. In this format, image pixels are bypassed layer by layer, gradually increasing the resolution [4, 30]. Therefore, long identical sequences of pixels are broken up and coded on different layers, which reduces CR. To reduce the manifestations of this deficiency in the following versions of the HBF-LS graphic format, we plan the following actions:

1. In the process of encoding, the end-to-end search between layers of identical sequences of the LZ77 algorithm from adjacent previously processed pixels is implemented. Such a modification should reduce CR.

2. During decoding, the array of pixels belonging to the same sequences from the previous layers should be matched with the unprocessed pixels of the image. This should reduce the amount of memory used.

VI. CONCLUSIONS

Therefore, applying the LZ77 algorithm for lossless image compression, in addition to the ideas given in [31-34], it is advisable to implement the following modifications:

1. Analyze and store bias to the same sequences not in the luminances of the pixel components, but in whole pixels. Determining the offsets to the same sequences of the LZ77 algorithm in pixels, and not in components, makes it possible to improve compression of images due to the encoding of three times smaller values and a threefold increase in the size of the dictionary.

2. Execute the search of the same sequences first of all starting from the closest previously processed pixels, and then in the dictionary. If the longest identical sequence is found, encode the offset to the nearest previously processed pixels with smaller values than the offset in the dictionary.

3. In the absence of the same sequence for buffer elements, instead of the next literal (the brightness of the pixel component), encode the result of using the predictor with the lowest entropy (as in the LZPR algorithm).

The application of such modifications and the LZPR algorithm provides, for example, a reduction in the CR of the ACT images by an average of 0.18 bpb.

In addition, to improve the compression of images in archivers, it is worth searching for identical sequences not only horizontally, but also vertically. This can significantly reduce the CR of images, although it will require the use of an additional array to indicate the belonging of pixels to previous replacements in the coding process.

References

[1] G. Wallace, "The JPEG still picture compression standard," *Communication of ACM*, vol. 34, pp. 30-44. 1991. https://doi.org/10.1145/103085.103089.

[2] J. Miano, *Compressed Image File Format: JPEG, PNG, GIF, XBM, BMP,* Addison Wesley, New York, 1999, 264 p. ISBN 0201604434.

[3] L. D. Crocker, "PNG: The portable network graphic format," *Dr. Dobb's Journal*. vol. 20, no. 232, pp. 36–44, 1995.

[4] A. V. Shportko, A. Ya. Bomba and V. A. Postolatii, "Programming the formation of difference color models for lossless image compression," *Proceedings of the 7th International Conference Computational Linguistics and Intelligent Systems (COLINS 2023)*, Kharkiv, Ukraine, Apr. 20-21, 2023, vol. 3, pp. 53-68. Available at: http://ceur-ws.org/Vol-3403/paper5.pdf.

[5] X. Li, M. T. Orchard, "Edge-directed prediction for lossless compression of natural images," *IEEE Transaction on Image Processing*, vol. 10, issue 6, pp. 813-817, 2001. https://doi.org/10.1109/83.923277.

[6] N. A. N. Azman, A. Samura, R. A. Rashid, F. A. Saparudin, M. A. Sarijari, "A hybrid predictive technique for lossless image compression," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1289-1296, 2019. https://doi.org/10.11591/eei.v8i4.1612.

[7] M. U. A. Ayoobkhan, E. Chikkannan, K. Ramakrishnan, S. B. Balasubramanian, "Prediction-based lossless image compression," *Proceedings of the International Conference on ISMAC in Computational Vision and Bio-Engineering 2018 (ISMAC-CVB)*, Springer, Cham, Jan. 02, 2019, vol. 30, pp. 1749–1761. https://doi.org/10.1007/978-3-030-00665-5_161.

[8] D. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, issue 9, pp. 1098-1101, 1952. https://doi.org/10.1109/JRPROC.1952.273898.

[9] A. Moffat, "Huffman coding," *ACM Computing Surveys (CSUR)*, vol. 52, pp. 1-35, 2019. https://doi.org/10.1145/3342555.

[10] J. Duda, K. Tahboub, N. J. Gadil and E. J. Delp, "The use of asymmetric numeral systems as an accurate replacement for Huffman coding," *Picture Coding Symposium*, Cairns, QLD, Australia, Jul. 30, 2015. https://doi.org/10.1109/PCS.2015.7170048.

[11] A. Gribov, "Optimal Compression of a Polyline with Segments and Arcs," 2017, 40 p. Available at: https://arxiv.org/pdf/1604.07476.pdf.

[12] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Journal of Research and Development*, vol. 20, issue 3, pp. 198–203, 1976. https://doi.org/10.1147/rd.203.0198.

[13] J. Rissanen, G. G. Langdon, "Arithmetic coding," *IBM Journal of Research and Development*, vol. 23, vol. 2, pp. 149–162, 1979. https://doi.org/10.1147/rd.232.0149.

[14] A. Moffat, R. M. Neal and I. H. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems*, vol. 16, vol. 3, pp. 256-294, 1998. https://doi.org/10.1145/290159.290162.

[15] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, issue 6, pp. 520–540, 1987. https://doi.org/10.1145/214762.214771.

[16] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 4, pp. 623-656, 1948. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x.

[17] Kh. B. Predko, M. V. Shovheniuk, "Canonical presentation of colour spaces for publishing systems," *Scientific Papers Ukrainian Academy of Printing*, vol. 2(16), 2009, pp. 60–73. Available at: http://nz.uad.lviv.ua/static/media/2-16/11.pdf.

[18] P. Deutsch, *DEFLATE Compressed Data Format Specification*, version 1.3, RFC 1951. Available at: https://www.rfc-editor.org/rfc/rfc1951.

[19] J. Ziv, A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, issue 3, pp. 337-343, 1977. https://doi.org/10.1109/TIT.1977.1055714.

[20] S. Kreft, G. Navarro, "LZ77-like compression with fast random access," *Proceedings of the 2010 IEEE Data Compression Conference*. Snowbird, UT, USA, Mar. 24-26, 2010, pp. 239-248. https://doi.org/10.1109/DCC.2010.29.

[21] H. Nagumo, M. Lu, K. Watson, "Image compression with modified LZ77 coding," *Proceedings of the IEEE Third International Conference on Signal Processing (ICSP)*, Beijing, China, Oct. 14-18, 1996, pp. 1067-1070. https://doi.org/10.1109/ICSIGP.1996.566277.

[22] M. Komar, A. Sachenko, V. Golovko and V. Dorosh, "Compression of network traffic parameters for detecting cyber attacks based on deep learning," *Proceedings of the IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, Ukraine, May 24-27, 2018, pp. 43-47. https://doi.org/10.1109/DESSERT.2018.8409096.

[23] P. Deutsch, J-L. Gailly, *ZLIB Compressed Data Format Specification*, version 3.3, RFC 1950, Network Working Group, 1996, 10 p. https://doi.org/10.17487/rfc1950.

[24] A. V. Shportko, *Rise of efficiency of compression of colored images in the PNG format. Thesis for a candidate's degree in technical sciences,* Rivne State University of the Humanities, Rivne, 2010, 195 p. Available at: https://dspace.megu.edu.ua:8443/jspui/handle/123456789/1665.

[25] A. V. Shportko, "Optimization of the use of static predictors in the process of lossless image compression," *Information extraction and processing*, vol. 28, 2008, pp. 82-89.

[26] W. K. Pratt, "Correlation techniques of image registration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-10, no. 3, pp. 353-358, 1974. https://doi.org/10.1109/TAES.1974.307828.

[27] *ACT – Test Files*, 2002, [Online]. Available at: http://www.compression.ca/act/act-files.html.

[28] *MinPNG 1.0 – a utility to minimize the size of PNG image files (True Color)*, 2010, [Online]. Available at: http://apserver.org.ua/peregl.php?d=view&tid=131.

[29] O. Shehata, "Unraveling the JPEG," *Parametric Press*, vol. 1, 2019. Available at: https://parametric.press/issue-01/unraveling-the-jpeg.

[30] A. Ya. Bomba, A. V. Shportko and V. A. Shportko, "Development of the HBF-LS graphics format for lossless progressive image compression," *Proceedings of the III scientific and technical conference Computational methods and systems of information transformation*, Lviv, Ukraine, Sept. 25-26, 2014, vol. 3, pp. 98-101.

[31] B. Rusyn, O. Lutsyk, Y. Lysak, A. Lukenyuk and L. Pohreliuk, "Lossless image compression in the remote sensing applications," *IEEE Proceedings First International Conference on Data Stream Mining & Processing (DSMP)*, Lviv, Ukraine, Aug. 1, 2016, pp. 195-198. https://doi.org/10.1109/DSMP.2016.7583539.

[32] H. D. Kotha, M. Tummanapally and V. K. Upadhyay, "Review on lossless compression techniques," *Journal of Physics,* vol. 1228, 2019. https://doi.org/10.1088/1742-6596/1228/1/012007.

[33] D. R. Pavithra, Sudarshan Patil Kulkarni, "Investigation of wavelets for representation and compression of skin cancer images," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 15, no. 2, pp. 24-34, 2023. https://doi.org/10.5815/ijigsp.2023.02.03.

[34] Z. Hu, O. Shkurat, K. Przystupa, O. Kochan, M. Ivakhnenko, "Low-light image enhancement technology based on image categorization, processing and retinex deep network," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 16, no. 5, pp. 1-13, 2024. https://doi.org/10.5815/ijigsp.2024.05.01.

Aleksandr Shportko, *Associate Professor of the Department of Information Systems and Computing Methods of the Private Higher Educational Institution "International University of Economics and Humanities named after Academician Stepan Demianchuk", Candidate of technical sciences. Graduated from Rivne state pedagogical Institute (1997) for specialty "Applied mathematics" and graduated from the school of Rivne state humanitarian university (2010).*

Scientific interests: compression of images without losses, theory of information, theoretical foundations of programming, design and development of databases.

e-mail: ITShportko@gmail.com, ORCID: 0000-0002-4013-3057

Andrii Bomba, *Professor of the Department of Computer Sciences and Applied Mathematics of the National University of Water and Environmental Engineering, Doctor of Technical Sciences, Candidate of Physical and Mathematical Sciences, Professor. Graduated from Lviv State University named after I. Franko (1972) majoring in "Mathematics", post-graduate studies at the Institute of Mathematics of the Academy of Sciences of the Ukrainian SSR (Kyiv, 1981), and doctoral studies at the Institute of Cybernetics named after V. M. Hlushkova, National Academy of Sciences of Ukraine (Kyiv; 2003).*

Scientific interests: basics of mathematical modeling, modeling of non-linear disturbances processes and systems, theory of information.

e-mail: abomba@ukr.net, ORCID: 0000-0001-5528-4192